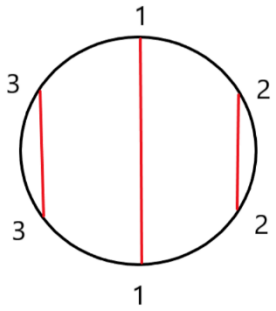
Object-Oriented Programming Language

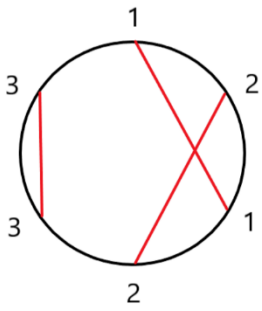
03/23/2023

Lab Assignment 5

1. Assume that n couples (labeled from 1 to n) are standing around a circle. They want to go on a date with their sweetheart. Albert, the god of marriage in NTUCE, is trying to red cords between the couples. The red cords might entangle with each other if they intersect. Thus, Albert needs to carefully check if there are any such intersections. Given n couple's position clockwise, please help Albert to check if the red cords intersect in O(n)-time.



[1,2,2,1,3,3]



[1,2,1,2,3,3]

For example, as shown above if the couples, labeled [1,2,2,1,3,3] respectively, are arranged clockwise on the circle, then Albert can draw between each two labeled 1,2 and 3, such that the red cords do not intersect. However, if the couples are labeled [1,2,1,2,3,3] respectively, the red cords 1 and 2 would intersect.

[hint: stack]

Sample runs of the program are as follows:

```
How many people ?
10
Please input all the number !
2 3 4 4 5 5 3 8 8 2
excellent!!
```

```
How many people ?
5
Please input all the number !
1 2 2 1 4
Oh no!!
```

2. Given an ordered deck of n cards numbered 1 to n with card 1 at the top and card n at the bottom. The following operation is performed as long as there are at least two cards in the deck:

Throw away the top card and move the card that is now on the top of the deck to the bottom of the deck.

Your task is to find the sequence of discarded cards and the last, remaining card.

Sample runs of the program is as follows:

```
How many card?
19
Discarded cards: 1 3 5 7 9 11 13 15 17 19 4 8 12 16 2 10 18 14
Remaining card: 6
```

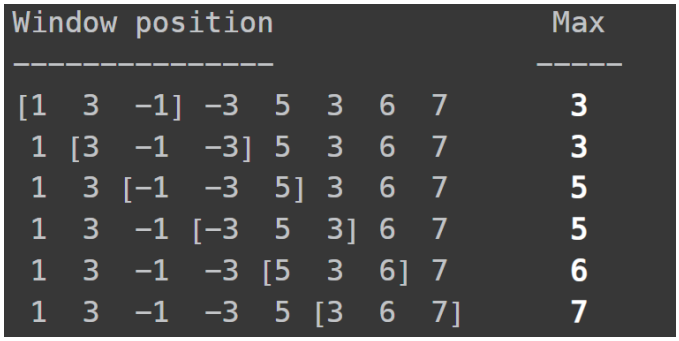
```
How many card?
4
Discarded cards: 1 3 2
Remaining card: 4
```

3.Write a program that inputs names and stores them in a map, then print out the count of every names

Sample runs of the program is as follows:

```
How many names?
5
Albert James Bert Albert Bert
Name: Albert, Value:  2
Name: Bert, Value:  2
Name: James, Value:  1
```

4.Given an array of integers with size of n. With a sliding window of size k moving from left of the array to the right, while each time the window moves right by one position, please print out the max integer of every windows, as picture shown below.



Window position	Max
[1 3 -1] -3 5 3 6 7	**3**
1 [3 -1 -3] 5 3 6 7	**3**
1 3 [-1 -3 5] 3 6 7	**5**
1 3 -1 [-3 5 3] 6 7	**5**
1 3 -1 -3 [5 3 6] 7	**6**
1 3 -1 -3 5 [3 6 7]	**7**

Here is some sample code that might be useful.

```
int main () {
    vector<int>nums,results;
    deque<int>q;
    int k,n,a;
    cin >> n;
    cin >> k;
    for(int i=0 ; i<n ; i++){
            cin>>a;
            nums.push_back(a);
    }
    for (int i=0; i<nums.size(); i++){
            while (q.size()>0 && nums[q.back()]<nums[i]){
                // Please fill this blank
            }
            // Please fill this blank
            if (q.front()<=i-k){
                // Please fill this blank
            }
            if (i>=k-1){
                // Please fill this blank
            }
    }
    for(auto &p : results){
        cout<<p<<" ";
    }
return 0;
}
```

Some sample runs of the program are as follows:
input: array size n, window size k, and array.
output: the largest number of each window.

```
8
3
3 4 5 2 1 8 3 9
5 5 5 8 8 9
```

```
6
3
3 2 1 4 7 9
3 4 7 9
```

5. Input an array of positive and negative numbers, find if there is a subarray (of size at least one) with 0 sum. If we try to consider all subarrays one by one, we need to run two loops. It takes $O(N^2)$-time to complete it.
Here is a special way that only use O(N)-time. That is using a hashing function and set. The idea is to traverse the array and maintain the sum of elements seen so far(that is your hashing function is the sum of array[i]). And check if the sum is seen before which means the sum exists in the set. (you can think why this work.)

```
Input:  { 3, 4, -7, 3, 1, 3, 1, -4, -2, -2 }

Output: Subarray with zero-sum exists

The subarrays with a sum of 0 are:

{ 3, 4, -7 }
{ 4, -7, 3 }
{ -7, 3, 1, 3 }
{ 3, 1, -4 }
{ 3, 1, 3, 1, -4, -2, -2 }
{ 3, 4, -7, 3, 1, 3, 1, -4, -2, -2 }
```

For example, as shown above. We can see the input array with so many subarrays with 0 sum.
Some sample runs of the program are as follows:

```
How many number in the array?
3
Input the array
1 2 -3
Found a subarray with 0 sum
```

```
How many number in the array?
7
Input the array
1 3 5 6 8 -2 -4
No Such Sub Array Exists!
```

```
How many number in the array?
5
Input the array
0 3 1 5 8
Found a subarray with 0 sum
```

6.Write a program that given n ropes of different lengths, connect them into a single rope with minimum cost. Assume that the cost to connect two ropes is the same as the sum of their lengths. The idea is to connect the two lowest-cost ropes first. The resultant rope has a cost equal to the sum of the connected ropes. Repeat the process (with resultant rope included) until we are left with a single rope. A priority queue implemented using min-heap is best suited for this problem.

For example, Input [5,4,2,8] the minimum cost is 36. Following is the process [5,4,2,8] ->[5,6,8] ->[11,8] ->[19] and the cost is sum of 4+2 ,6+5 , 11+8 , that is 6+11+19=36

Some sample runs of the program are as follows:

```
Input your length of every ropes, end  with 'q'
5 4 2 8 q
The minimum cost is 36
```

```
Input your length of every ropes, end  with 'q'
1 1 6 3 4 8 8 q
The minimum cost is 78
```

7.Write a program that reverse specific portion of a list. You should only use list

You cannot use string, vector, array or other.

Some sample runs of the program are as follows:

```
Where to reverse the list? from m to n
2 7
Input your list, end  with 'q'
1 2 3 4 5 6 7 8 9 10 q
After reverse
1 7 6 5 4 3 2 8 9 10
```

```
Where to reverse the list? from m to n
3 9
Input your list, end  with 'q'
55 33 454 1212 62220 5454 1 54 64 87 99
q
After reverse
55 33 64 54 1 5454 62220 1212 454 87 99
```

8. Write a program that find the previous smaller element for each array element. In other words, for each element A[i] in the array A, find an element A[j] such that j < i and A[j] < A[i] and value of j should be as maximum as possible. If the previous smaller element doesn't in the array for any element, consider it -1.

The time complexity of brute force approach is $O(N^2)$. Traversal all the left element for every element. Try to use stack to approach time complexity O(N).

(Note that the first element always has a previous smaller element as -1.)

Some sample runs of the program are as follows:

```
Please input array, end  with 'q'
2 5 3 7 8 1 9 q
-1 2 2 3 7 -1 1
```

```
Please input array, end  with 'q'
5 7 4 9 8 10 q
-1 5 -1 4 4 8
```